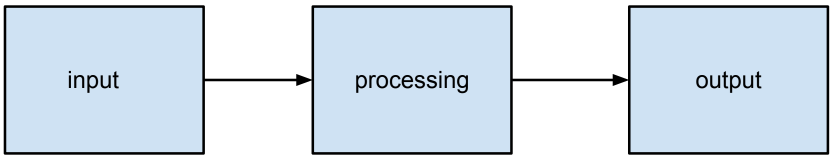
TSBB15
Computer Vision

The Invisible Pixles

Object tracking



Benjamin Ingberg
Li Sandsveden
Malin Rudin
Mattias Nilsson
Patrik Hillgren

April 7, 2013

Preamble

We would like to thank Erik Ringaby for excellent guidance, recommendations and help.

Contents

1 Overview

This report will present the final results of the object tracking project in the course TSBB15 Computer Vision at Linköping University, spring semester 2013.

During the project we worked very much together as a team so there was always more than one student involved in every single part of the project. In the beginning we did however divide the project into different parts for which we had a main responsibility each. The responsibility does not mean that it was the one responsible who did all the work. The areas of responsibility was:

- Background modelling - Malin
- Shadows and reflections - Li
- Visualization, prediction and evaluation - Mattias
- Object Identity Management - Patrik
- Occlusions - Benjamin

1.1 Problem description

The assignment was to implement a tracking system that tracks objects in a simple sequence. Object identity should be handled and the result should be evaluated. In addition, the system should have functionality to handle shadows, reflections and occlusions. It should also have more than one background model to handle spurious motions i.e. small movements that are not from interesting objects.

1.2 System overview

The system consists of subsystems that handle background modelling, shadow and reflection detection, segmentation, object identity and visualisation. There is also an evaluation subsystem used to evaluate how good the result is. In figure 1 the overall system is illustrated.

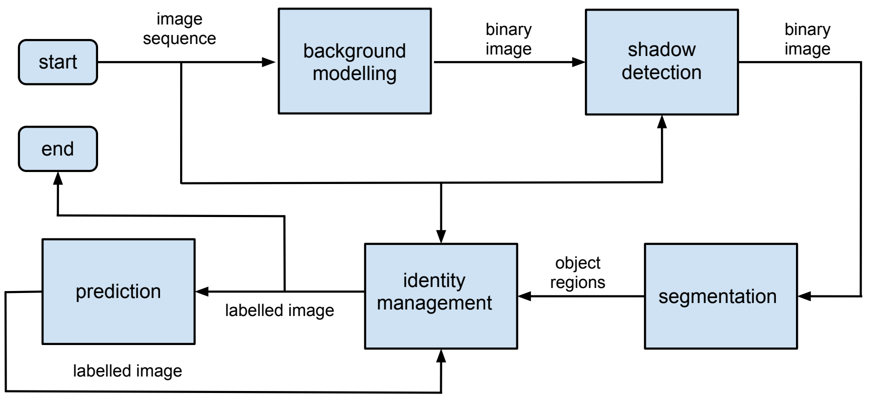


Figure 1: An overview block diagram

2 Implementation

The system has been implemented in C++, consisting of different parts that together make up the whole system. The system is started from an executable file that takes a text file containing parameters as input. After starting the program the file type and path to the sequence is specified.

The program then displays three black and white images and the final result for each frame. The black and white images shows the results of the background model, the shadow detection and the segmentation preprocessing respectively. The final result consists of the original image with boxes around interesting objects.

The different parts of the program and their functionality are described below.

2.1 Background modelling

The first step in the tracking system is to estimate a background model. The idea is to have the interesting objects separated from the background which is done by individually examining each pixel and determining whether it is part of the background or not. A proper background model is crucial for a good result.

The background model is represented by a binary image where the value one represent the foreground and the value zero represent the background. There are some different approaches to how this model is estimated and given the requirements for this tracking system some are more suitable than others.

2.1.1 Mixture of Gaussians

The image sequences used in this project include spurious motion which a simple background model can not handle since it would classify it as foreground. For this tracking program the choice of model was a mixture of Gaussian distributions. This approach allows for a multi-modal adaptive background model, that is capable of removing spurious motions.

The basic principle is to model each pixel as a mixture of K Gaussians, where some of the Gaussians represent background and some represent foreground. To estimate the mean μ_k, covariance σ_k^2 and mixing densities w_k for these distributions an expectation-maximization algorithm is used [5].

2.1.2 Algorithm description

The algorithm used to implement the mixture of Gaussians is based on the algorithm and theory recommended by John Wood in his master thesis [2].

The first step in the algorithm is to determine which of the K distributions generated the pixel value through calculating the Mahalanobian distance over the different color channels:

$$d_k^2 = \sum_{d=1}^{D} \frac{(x_{t,d} - \mu_k, d)^2}{\sigma_{k,d}^2} \tag{1}$$

If $d_k < \lambda$, the distribution is considered a match for the pixel and the parameters are updated. If several of the distributions matches, the one with the highest peak will be chosen. This will always be the first one since the distributions are sorted by their peak $\frac{w_k}{||\Sigma_k||}$, where $\Sigma_k = \sigma_k^2\mathbf{I}$. If no distribution matches, the lowest peaking distribution is replaced by a new distribution with the current pixel value as its mean, large initial values along the diagonal of the covariance matrix and the learning rate of the system as its mixing density. If a match was found, the mean, covariance and weights of that distribution are updated according to:

$$w_k := (1-\alpha)w_m + \alpha \tag{2}$$

$$\rho := \frac{\alpha}{w_m} \tag{3}$$

$$\mu_k := (1-\rho)\mu_k + \rho \cdot x \tag{4}$$

$$\sigma_k^2 := (1-\rho)\sigma_k^2 + \rho(x-\mu_k)^2 \tag{5}$$

Where α is the learning rate of the system and x is the current pixel. After that the distributions are resorted by their peak.

To avoid the algorithm to be too sensitive to small changes a lower limit on the covariance is used. Since each match will reduce the covariance of a pixel the background model would be too unforgiving if giving after a large amount of iterations and introduce a lot of noise.

The first distributions to have a combined mixing density greater than the threshold T are considered background. If $d_k < \lambda$ holds for any of these distribution, the pixel is considered background. Figure 2 shows an example of the binary background model image and the corresponding result image.

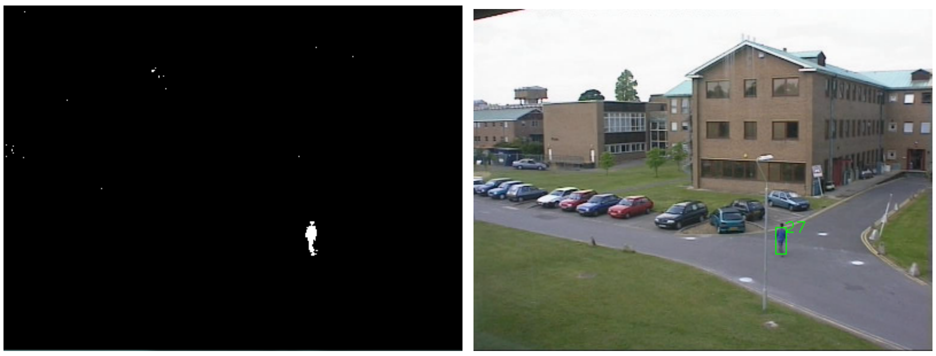


Figure 2: Binary image of the background model and corresponding result image

2.1.3 Parameters

The background model takes an image as input and returns a binary image as output. It also takes some parameters as input, which are the learning rate, the combined background density threshold, the matching threshold, the minimum standard deviation, the initial standard deviation and the number of mixture components.

The learning rate, α, defines how fast the background model is updated. The match threshold, λ, is used to decide if a pixel matches a model. The combined mixing density threshold, T, is used to tell which models that are background and which that are foreground.

The number of mixture components, K, is the number of Gaussian models used. The initial standard deviation, σ_{init}, is used as a starting value for the model if no distribution is matched to the pixel. The minimum covariance, σ^2_{min}, defines the smallest value allowed for the covariance for the model.

The choice of parameters will affect the outcome of the background model and different image sequences will need different parameters. To get an idea of which values that are reasonable to start with, the master thesis of John Wood [2] was consulted and occular evaluation of the result was used to improve the values for specific sequences.

To get a good estimate of the background model, the program needs to run for a while. At first there will be no foreground, since the program will consider all pixel values to be within the large initial variance of the models. Depending on which learning rate is chosen the number of required frames will vary. In general 50 frames will be sufficient for the learning rates chosen in the project.

2.2 Shadow detection

The background model will classify shadows and reflections of objects as foreground. To handle this a shadow detection algorithm is used. The algorithm was first proposed by Horprasert et al in 1999 [1] and is based on a computational color model which separates the brightness and the chromaticity of a pixel.

2.2.1 Brightness and Chromaticity

The principle of the shadow detection algorithm is to find which parts of the foreground that are shadows. The difference between shadows and objects are that the shadows are transparent and less bright. This can be measured with the chromaticity and brightness component described in figure 3.

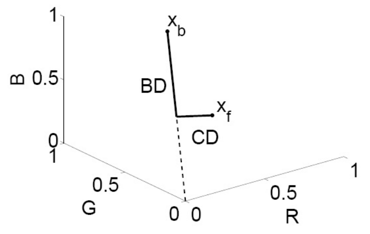


Figure 3: Brigthness and chromaticity distortion (BD and CD) in the RGB color space. x_b and x_f denotes background and foreground values respectively. Figure from John Wood [2].

In the RGB space every point represents a color. Movement along a line through the origin and a specific point represents brightness distortion. Moving closer to the origin lowers the brightness. Perpendicular movement to that line represents chromaticity distortion. Moving away from that line means higher chromaticity distortion.

2.2.2 Algorithm description

For shadow detection the recommended algorithm from the master thesis by John Wood [2] was used. A block diagram of the implemented algorithm is shown in figure 4.

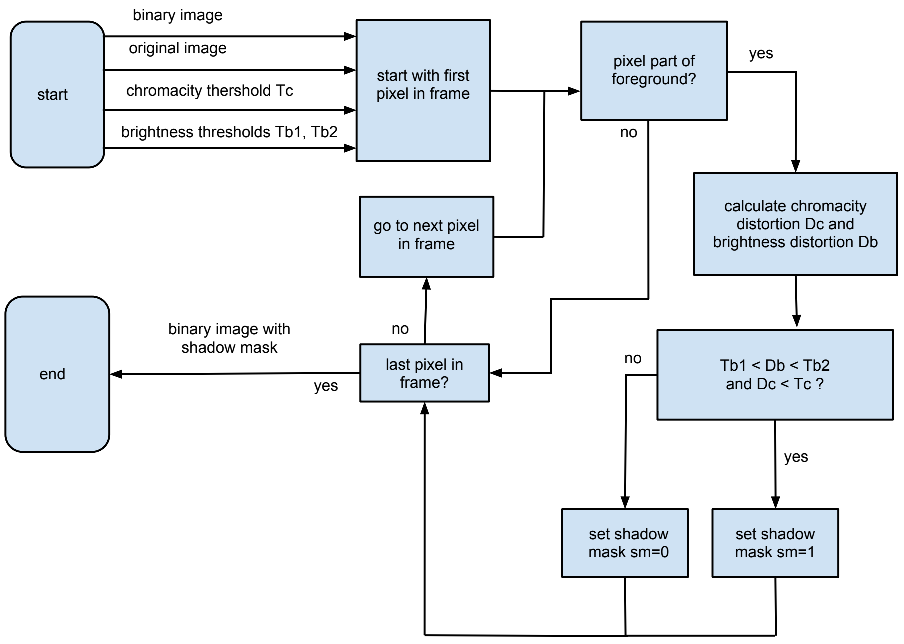

binary image
original image
chromacity thershold Tc
brightness thresholds Tb1, Tb2
start
start with first pixel in frame
pixel part of foreground?
yes
no
calculate chromacity distortion Dc and brightness distortion Db
go to next pixel in frame
end
binary image with shadow mask
no
last pixel in frame?
yes
no
Tb1 < Db < Tb2 and Dc < Tc ?
yes
set shadow mask sm=0
set shadow mask sm=1

Figure 4: The implementation of shadow reduction

As seen in the figure the input to the subsystem will be a binary image as well as the original image. The output will be an updated binary image where the shadows and reflections have been added to the background. In figure 3 this is seen as setting the parameter shadow mask, *sm*, to one.

2.2.3 Parameters

The input parameters that controls how strong the shadow detection will be are three thresholds, which can all be chosen by the user. This is necessary since different values will be optimal for different image sequences.

The three thresholds are Tc, that controls the maximum chromaticity distortion, TbHigh and TbLow, that describes an interval for acceptable brightness distortion. The result differ significantly depending on how the parameters are chosen. The following figures show two examples. In figure 5 parts of the shadow is not detected. To generate this image TbLow was set to 0.05, TbHigh was set to 0.95 and Tc was set to 0.05. In figure 6 too much of the object is considered shadow. As can be seen, a part of the woman's legs and hair is wrongly considered shadow. To generate this image TbLow and TbHigh was still 0.05 and 0.95 respectively but Tc was 0.2.

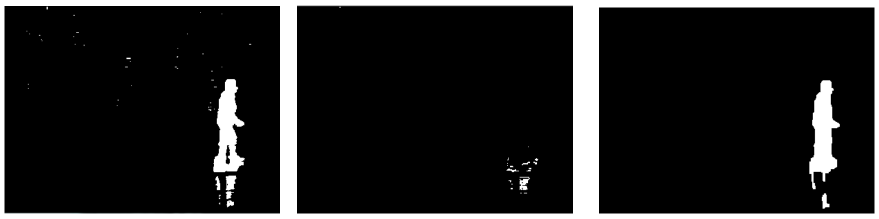

(a) Detected foreground (b) Detected shadows (c) Resulting objects

Figure 5: Too weak shadow detection

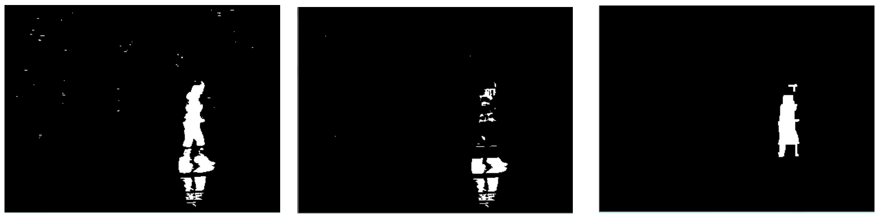

(a) Detected foreground (b) Detected shadows (c) Resulting objects

Figure 6: Too strong shadow detection

2.3 Segmentation

The segmentation subsystem transforms the binary image into regions which are candidates for objects. The binary image result cannot be immediately used for this, partially because there will be pixels that are falsely classified.

To transform pixel groups into regions the segmentation subsystem does preprocessing with morphological operations. After preprocessing it performs labelling with the two-pass algorithm [10] and removal of small objects by thresholding the distance between the centerpoint of each object and the nearest boarder.

2.3.1 Morphological operations

Since the binary image does not present a perfect result some unwanted features have to be removed. One problem with the binary image is that the whole object is not found. Another problem is that there is noise in the binary image.

To decrease the effects of these artefacts the morphological operations opening and closing are used. The operation closing is used to expand the pixels that are found by the background model, i.e the holes that might exist in objects are filled. The operation opening is used to contract the objects found, i.e noise is removed. The strength and order of openings and closings are parameters defined by the user and is often different for each video sequence. [8]

2.3.2 Labelling and removing of small objects

To be able to return regions containing objects each region is labelled with a number. This is done by an implementation of a connected-component labelling algorithm. The algorithm implemented in this project is the two-pass algorithm [10] which assign each non-zero pixel a label and merges connected pixels to one object with the same label.

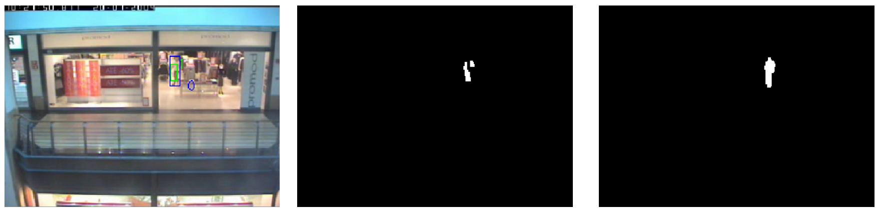
Too small objects will be removed because of the probability that they are just noise. To be able to remove smaller objects in the image the distance from each objects center point to a zero valued pixel is calculated. This is done by using the method distImage already implemented in OpenCV [6]. There is a threshold, set by the user, for an object to be considered as large enough. This method has some advantages in comparison to more simple methods such as only considering the total number of pixels for each object. For instance if there exists a very long but thin object in a frame this would be considered as an object if the number of pixels only would be considered, however not by the distance transform.

The objects that are left after removing small objects are considered as objects by the program. The segmentation returns the two points in the image corresponding to the top-left corner and the bottom-right corner of each object. These two point are found by first calculating the contour points of each object and finding the ones that corresponds to the smallest/largest coordinates. The contours are found by using findCountours in openCV [6]. The regions that are found are passed on to the object identity subsystem. Which finds out which region corresponds to which between frames and that makes decision whether an object should be considered as a true object or not.

2.3.3 Parameters

The input to the segmentation is a binary image corresponding to the background model with shading and reflections removed. The parameters that are used in the segmentation is the number of dilations and openings and the threshold for the distance used for removing small objects. The choice of parameters in the segmentation will affect the result significantly, since the segmentation decides where the objects should be detected and drawn.

The results of one segmentation of one frame of the segmentation is presented in 8. To illustrate the effect of a proper parameter choice there are two output images corresponding to the same input image, which have different opening strengths. As you may notice there is a major difference in the performance of the segmentation by only adjusting one of the parameters.



(a) Original image (b) 1 Closing (c) 4 Closing

Figure 7: Input image and corresponding output images after segmentation pre-processing

2.4 Object identity

Each region returned by the segmentation belongs to a certain object. The object identity subsystem matches regions to already existing objects, introduces new objects and removes old objects.

2.4.1 Visibility counter

To handle the sporadic nature of the objects each object has a visibility counter. The visibility counter is incremented for every frame where the object is detected and decremented for every frame where it is not detected.

Only objects that has a visibility counter above a certain threshold are considered true objects and visualized, they are visualised as observed if they were detected in the current frame and as occluded if they were not. In addition there is a maximum allowed value on the visibility counter so that long term objects are allowed to disappear in reasonable time while still giving them preference above short term objects.

2.4.2 Matching

Using the earlier results a prediction of the next result for each object can be calculated, which will be discussed more in the next section. The predictions needs to be matched with the new regions.

A prediction is considered to match with the region that has the smallest weighted square below a certain threshold. Since objects entering into each other would be identified as one object by the segmentation algorithms a region can match to several predictions. However several regions can not match to the same prediction.

The error weights allows a steeper penalty to be set for different kinds of movement. In a scene where objects move fast on the horizontal axis but slowly on the vertical axis a higher penalty on vertical movement would result in better matching. These parameters are dependent on the environment being simulated.

The matching system leaves two different problems, predictions that aren't matched to any region and regions that aren't matched to a prediction.

Predictions which are not matched to any region are considered occluded. Occluded regions are only visible if their visibility counter is sufficiently high. This allows objects to not be detected for a few frames. After the object is no longer visualized it is kept for another couple of frames before it is discarded.

If a region does not match a prediction it will be added as a new object with a visibility counter of zero. This allows new object to enter the scene as well as one object splitting into several objects.

Since several predictions can be matched towards the same region objects has a possibility of becoming very similar if they do not separate in a timely manner. To handle this situation very similar objects are considered to be the same object and merged into the oldest object. This feature diminishes the number of objects created by noise and also allows two previously separated objects to act as one coherent object.

2.4.3 Prediction

Based on the movement of an object the next position can be predicted with some accuracy. For this Kalman filtering is used since it is robust to both noise and model errors.

As a model for the Kalman filters the position and size of the region as well as their time derivatives is used. This is a fairly simple model with few parameters and the Kalman filtering performs well on it. Since the time derivatives of the position and size are not known they are interpolated by the Kalman filters.

2.4.4 Parameters

The object identity subsystem has several parameters. While the model used by the Kalman filters could be considered as several parameters they were hard coded for this project.

The parameters for the subsystem are the weights for each of the errors, width, height, x- and y-position of the matching, the maximum weighted square error to be considered a match, the minimum amount on the visibility counter for the object to be visible and the maximum allowed value on the visibility counter.

2.5 Visualization

The primary target of the project is to show videos with drawn rectangles around every labelled object. The objects also have unique numbers showing that the system knows which object is which. The program also prints the correct values given on the course web-page so that is easy to see how well the program performs. The Visualization uses different colors depending on what it is showing. A green rectangle means that it shows an object the system has found. A red rectangle means that the system is showing an occluded object that is predicted by the Kalman filter. A blue rectangle means that the system is showing ground truth for an object.

2.6 Evaluation

Evaluating a system that give visual results is not intuitive. It is easy to see if it works well but it is hard to put actual numbers on its performance.

In this project two different evaluation methods are used to analyse and evaluate the system. The first method is called MOTA and evaluates how often the system finds correct objects. The second method is called MOTP and evaluates how well the system approximates the object it finds. MOTA and MOTP are described in more details in [4]. Both methods assume that you have access to ground truth values for the video sequences you want to analyse.

2.6.1 Ground Truth

To evaluate the system datasets from CAVIAR Test Case Scenario [7] was used. These datasets comes with a *Ground Truth*. This ground truth is the correct tracking of the objects in the video sequences and are regions in each frame where moving objects are found. These regions have been found manually by the CAVIAR group.

2.6.2 MOTA

MOTA stands for Multiple Object Tracking Accuracy and evaluates how often the system finds correct objects. The parameters used in MOTA and the MOTA-equation itself is listed below.

T - threshold value. Since there is more than one group for this project a common threshold value is used. That value is 20.

m_f - number of misses at frame f. A miss occurs if the system is not placing an object closer than T to a ground truth object. The distance is measured euclidean from the center of the estimated object to the center of the ground truth object.

mm_f - number of mismatches at frame f. Miss matches is when the system accidentally switches identity of two objects.

fp_f - false positives at frame f. A false positive is when the system puts an object where there is no ground truth object. Note that an object can not be both a miss and a false positive.

g_f - the number of ground truth objects at frame f.

$$MOTA = 1 - \frac{\sum_f m_f + mm_f + fp_f}{\sum_f g_f} \tag{6}$$

MOTA will typically give values in the range [0, 1] where a perfect system gets the value 1 and a completely non working system gets 0. Negative values are theoretically possible in the case of a large amount of false positives but for a working system MOTA can be considered as a correctness measurement from zero to one. A system that has a negative MOTA value is really bad since it generates more errors than there are correct objects.

2.6.3 MOTP

MOTP stands for Multiple Object Tracking Precision and evaluates how precise the system finds the correct objects when it finds them. The parameters and the equation of MOTP are listed below.

d_{if} - The distance between the objects in a matching pair i of a ground truth object and an object estimated by the system, for a frame f.

c_f - the number of correctly matched objects in a frame.

$$MOTP = \frac{\sum_f \sum_i d_{fi}}{\sum_f c_f} \tag{7}$$

2.6.4 Matching

A problem when using both MOTP and MOTA is the matching. Even if an object is the closest to a reference there might be a more optimal matching. To solve this problem optimally the Hungarian assignment algorithm is used. [9]

2.6.5 Evaluating subsystems

To get a good indication of how a system works it is important to analyse the subsystems individually. That way it is easier to say which part that could use improvement. The evaluation can be done both using hard values such as MOTA and MOTP and with soft values as how the visual result looks. Evaluating using hard values such as MOTP and MOTA is important to make a system that is comparable to other systems. But evaluating a system using soft values as visualization is also good since humans can see and interpret what happens in an image. Using only hard value evaluation it would be hard to know what to change if the hard values are bad.

In the beginning of the project the plan was to have artificial tests for every part of the system. This was done for some parts of the project such as Visualization, Segmentation and Object Prediction but it was not done for Background modelling and Shadow detection. The reason Background modelling was not tested with artificial tests is that it needs an image sequence to work. Hence it is easier to run it on a real image sequence than creating a new one. The Visualization subsystem was tested using artificial moving objects that were visualized in a window on the screen. The Segmentation subsystem was tested using an artificial binary image created in MS Paint. The Object Prediction part was tested by letting an artificial object move in a ring. Then the estimated value and the predicted value for next frame was calculated by a Kalman filter and visualized by Visualization.

To make the soft value evaluation easier the system always visualizes all the steps in the program. A window is plotted for background modelling, shadows removed by the system, after segmentation preprocessing and a visualization of the final result.

One method to hard value evaluate the different parts of the system is to disconnect them and see how MOTA and MOTP is affected. This evaluation can not be done for all parts of the system since some parts are essential for the program to work. Parts that were evaluated this way was Shadow Detection and Object Recognition. It would have been better to also hard value evaluate the Background modelling but since there was no way of getting ground truth values of how it should work it was not possible. For Background modelling ground truth would be a perfect binary image that had the background equal to zero and every object equal to one.

The results of the Evaluation are shown in section 3 and discussed in section 4.

3 Result

The result is presented in soft and hard values. Soft values means that we evaluate the result in a subjective way, here images of the result will be presented. Hard values are the result from the evaluation in numbers. In the result below four different image sequences are analysed. All the videos are captured at the same scene and therefore the same parameters are used for all of them. The videos can be found at [7]. It was chosen not to optimize parameters more than necessary because it would be too time consuming and is considered as not in the scope of the course. The parameter values were, for the parameters that it were possible, for starters taken from John Wood[2]. The other parameters were chosen by intuition. Approximately one hour was then spent tuning the parameters by hand.

3.1 Image Sequences

OneLeaveShop1ByFront.mpg - A person walks out of a shop. Some other persons can sometimes be seen in the background. The video should be quite easy to track except that he is moving when the video starts and the persons in the background can be hard to find.

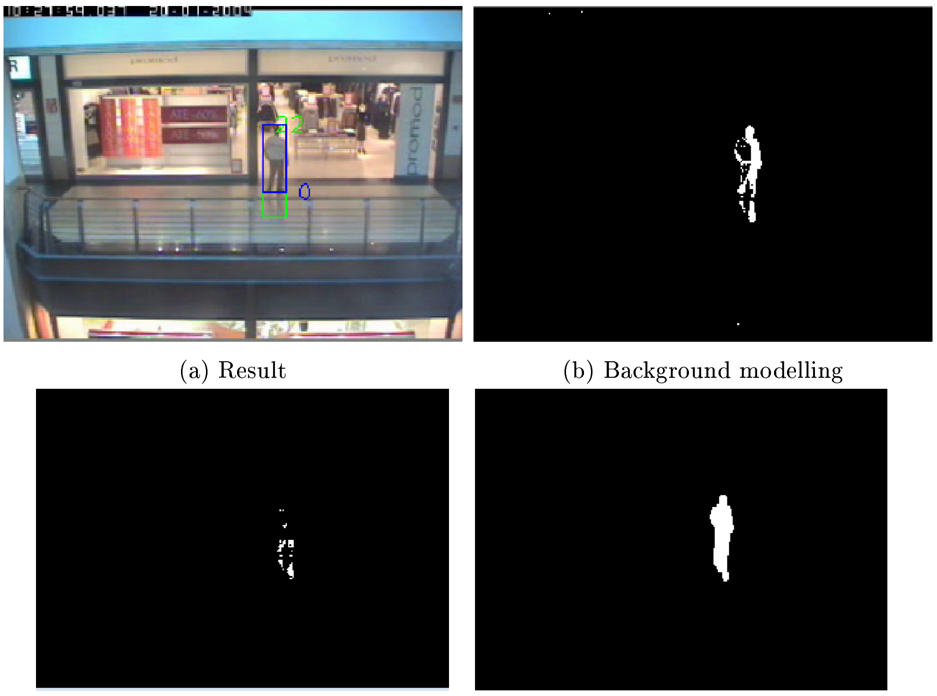
OneLeaveShop2Front.mpg - A person walks out of a shop. The image sequence is similar to OneLeaveShop1Frong.mpg but there is none in the background.

OneLeaveShop2ByFrontReenter.mpg - A person walks out of the scene and comes back again. Another person also walks by. Probably an easy video to track.

WalkByFront1front.mpg - The scene is zoomed in. A person walks by really fast. The video is tough to track since the person moves so fast.

3.2 Soft evaluation

In general the system works well and often recognizes moving objects. However we often find only parts of the ground truth which leads to poor values in the MOTA and MOTP calculations. This means that the soft evaluation is useful to find out if the program is working in any way and to find areas of improvement. One example where soft evaluation is better than hard evaluation is when considering shadows. If a shadow is detected which removes parts of the real object a major problems might occur. This means that the center point of the estimated object will be moved and it might be seen as a false positive by our other evaluation methods. Another example is that it is visibly possible to notice when two objects are close and melt together in the labelling. This is hard to show using hard values. Below are 4 images showing the results of one frame.



(a) Result (b) Background modelling

(c) shadows removed (d) labelled image

Figure 8: Input image and corresponding output images after segmentation pre-processing

3.3 Hard values

Table 1: MOTA and MOTP results

Video	MOTA	MOTP
OneLeaveShop1ByFront.mpg	0.42	4.14
OneLeaveShop2Front.mpg	0.54	5.25
OneLeaveShop2ByFrontReenter.mpg	0.64	5.76
WalkByFront1front.mpg	0.13	6.90

The results may seem pretty bad but the visual evaluation of 0.64 MOTA is almost perfect. Most of the errors occur from small objects that are barely visible. A really good MOTA value is received when there is an interval in the beginning of the video so that the system is able to adapt. MOTP is in general small i.e. the system is quite precise. Below are the parameters used for the videos.

Table 2: Background modelling parameters

Parameter	Value
K	3
σ_{init}	0.08
σ^2_{min}	0.04
α	0.015
λ	2.5
T	0.8

Table 3: Segmentation parameters

Parameter	Value
open Before close	true
Closing strength	4
Opening strength	1
Depth threshold	1

Table 4: Shadow detection parameters

Parameter	Value
Tc	0.04
Tblow	0.3
TbHigh	0.95

Table 5: Object prediction parameters

Parameter	Value
Maximum Error	30
Maximum Visibility Counter	10
Minimum Visibility Counter	3

4 Discussion

A discussion about the result, parameters and possible improvements follows below.

4.1 Result

Greater parts of our implementation is based on existing algorithms and proven methods and hence the most critical design choice would be the choice of parameter values. The parts that are not based on a given algorithm, i.e. the segmentation, the object identity management and the occlusion, have been developed by the project group. These parts could be considered less reliable and another design choice could have given other, and maybe better, results. Also, for these parts no parameter values are known and because of that the choice of parameter values for these parts might be less reliable.

We believe that the results are sufficient. The objects of interests are most often identified even though the framing of them might be a little of. This is most likely due to the choice of parameter values since this choice is not trivial. Also, a perfect tracker is more or less impossible to obtain even tough this was what we aimed for.

4.2 Parameters

The system uses more than twenty different parameters. These parameters are an essential part of what result we will get. It would have been nice to have an automatic way to find values for the parameters. Since the parameters are not truly independent of each other parameter tuning is a significant part of running the program.

For similar scenes parameters can be mostly left alone.

4.3 Possible Improvements

Since the project was limited in scope (two work weeks) our implementation is very superficial. Had the project been longer there are several things that could have improved the system. Below will follow a list of a few improvements that are outside of the scope of our project.

4.3.1 Dynamic parameters

Certain parameters are not very well described as constants and could use scaling with other parameters. For example an error of 20 pixels is much less significant in a 1920x1080 sequence than in a 320x200 sequence.

4.3.2 Machine learning

Certain parameters could have been automatically identified using unsupervised machine learning. Even eliminating a few of the parameters would have had a profound effect on the difficulty of parametrizing the system since the parameter space grows exponentially with the number of parameters.

Using supervised learning all parameters could have been identified using the evaluation results, i.e. minimizing MOTA and MOTP. In practice this could be useful for stationary cameras with a known environment that does not change very often.

4.3.3 Prediction and matching

Currently the matching algorithm uses weighted square error between the region and the prediction. However a more suitable error could have been calculated by using maximum likelihood from the estimated covariance matrix of the Kalman filter.

The matching algorithm could have in turn been replaced by the Hungarian maximum gain bipartite matching used by the evaluation. This would however require completely new behaviour to eliminate false positives, occlusions and merges.

The Kalman filter model could have been replaced by a projection of a flat 3D-world into a 2D-plane. This however would have greatly increased the number of parameters for the model. The model could also itself be considered parameters of the system.

4.3.4 Segmentation

The segmentation requires both opening and closing to create coherent objects. An alternative would have been to use the shrinking to skeleton method.

4.4 Evaluation methods

As mentioned in the Evaluation section 2.6 it is hard to come up with good evaluation methods for systems with visual results. MOTA and MOTP are good methods since they determine both the accuracy and the precision of the system. MOTP does not determine the precision perfectly though. The method does not take account for the size of the objects it is comparing. If one of the objects is 100 times bigger than the object it is compared to but their midpoint is at the same place they will be considered a perfect match. MOTA determines a match in the same way, thus it has the same size problem. However, since it measures the accuracy and not the precision, it is not as crucial. MOTA is a very good evaluation method but MOTP could use an adaptation.

One issue for our use of MOTA and MOTP is that several videos starts with immediate motion while our background model requires some time to determine the variance of the background. Thus early objects gives large amounts of errors and gives better MOTA and MOTP result for high learning rates and low initial standard deviation while making the overall result worse.

5 External Libraries

Several external libraries are used in this project. Below is a short description of the external libraries used and which functionality they have served in the implementation.

5.1 RapidXml

To be able to evaluate the performance of the program the results was compared with the ground truth of the videos sequences, i.e the correct extraction of moving objects in the scene. The ground truth found in the set CAVIAR [7] have the ground truth expressed in XML files. To be able to extract the ground truth we used the external library RapidXml which makes it fairly simple to read XML files. RapidXml makes it possible to traverse an XML tree and find the ground truth. An external library was used because it saved time.

5.2 OpenCV

OpenCv [6] is an external library for image processing. Below are the functionalities from OpenCV, which are used in our implementation, listed.

5.2.1 DistanceTransform

The function DistanceTransform calculates the distance from each non-zero pixel to a zero valued pixel.

5.2.2 Erode

The function Erode performs the morphological operation erosion, which shrinks the objects in the image. Which pixels that are set to zero is decided by the structuring element that the operation is using.

5.2.3 Dilate

The function Dilate performs the morphological operation dilation, which expands the objects in the image. Which pixels that are set to one is decided by the structuring element that the operation is using.

5.2.4 FindContours

The function FindContours finds the position of the pixels that are in contact with a zero valued pixel in an image. This is used to find the pixel positions of the two pixels that will enclose the object.

5.2.5 KalmanFilter

The KalmanFilter implementation from OpenCV is used by the prediction system.

References

[1] T. Horprasert, D. Harwood, and L.S. Davis *A statistical approach for real-time robust background subtraction and shadow detection.* In Proceedings of IEEE ICCV'99 FRAME-RATE Workshop, 1999.

[2] John Wood *Statistical Background Models with Shadow Detection for Video Based Tracking.* Linköpings universitet, Institutionen för systemteknik, Bildbehandling, 2007.

[3] Course homepage for TSBB15, found at URL: http://www.cvl.isy.liu.se/education/undergraduate/tsbb15/object-tracking-project

[4] Karl Bernardin, Alexander Elbs, Rainer Stiefelhagen *Multiple Object Tracking Performance Metrics and Evaluation in a Smart Room Environment.* Universität Karlsruhe, Interactive Systems Lab

[5] M. Sonka, V. Hlavac and R. Boyle *Image Processing, Analysis, and Machine Vision.* 3rd edition, Thomson, 2008.

[6] Open CV - Open Source Computer Vision, homepage found at URL: http://opencv.org/

[7] EC Funded CAVIAR project/IST 2001 37540 found at URL: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/

[8] R. Gonzales and R. Woods *Digital Image Processing* 3rd edition, Pearson Education, Inc., 2008.

[9] A high level description of the Hungarian algorithm can be found at http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=hungarianAlgorithm

[10] Wikipedia, Connected-component labeling, found at URL: http://en.wikipedia.org/wiki/Connected-component_labeling